Modern Feynman Diagrammatic One-Loop Calculations with Golem, Samurai & Co.

Thomas Reiter (Nikhef)

in collaboration with

G. Cullen, N. Greiner, A. Guffanti, J.P. Guillet, G. Heinrich, S. Karg,
N. Kauer, T. Kleinschmidt, E. Pilon, M. Rodgers, I. Wigmore

and

P. Mastrolia, G. Ossola, F. Tramontano

and

M. Koch-Janusz

CPP2010, KEK, Tsukuba, 23–25 September 2010

Overview

Motivation

Improved Tensor Reduction (Golem95)

Reduction at the Integrand Level (SAMURAI)

Tensorial Reconstruction at the Integrand Level

Assembling the Golem (`golem-2.0`)

Results for $q\bar{q} \to b\bar{b}b\bar{b}$

Outlook and Summary

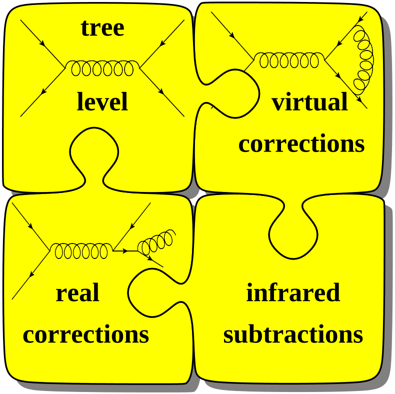
Motivation

Multi-leg NLO calculations are not feasable with traditional reduction techniques.

On-shell methods have shown to be efficient in QCD calculations with many external partons.
In BSM calculations Feynman diagrams still are indispensable.

How can we use Feynman diagrams most efficiently?

Anatomy of an NLO Calculation

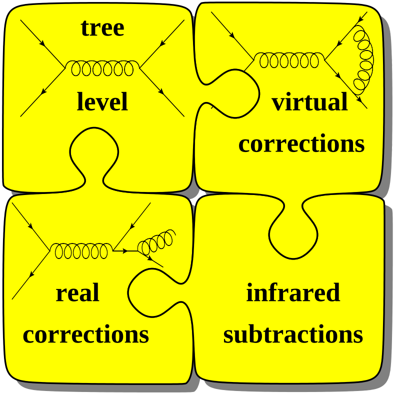


Complete NLO amplitude requires

- tree level amplitude,
- virtual corrections,
- real emission,
- subtraction terms
- Monte Carlo integrator, parton shower, PDFs, ...
- ... *but* the problem is nicely modular.

Golem and Samurai collaborations focus on virtual corrections

Anatomy of an NLO Calculation

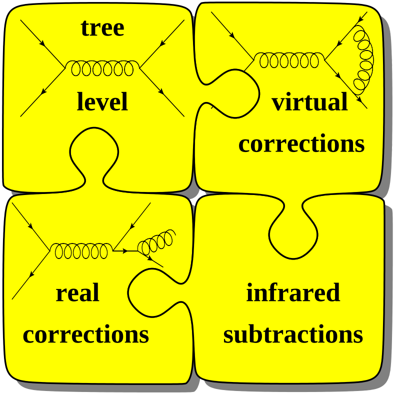


Complete NLO amplitude requires

- tree level amplitude,
- virtual corrections,
- real emission,
- subtraction terms
- Monte Carlo integrator, parton shower, PDFs, . . .
- . . . *but* the problem is nicely modular.

Golem and Samurai collaborations focus on virtual corrections

Anatomy of an NLO Calculation

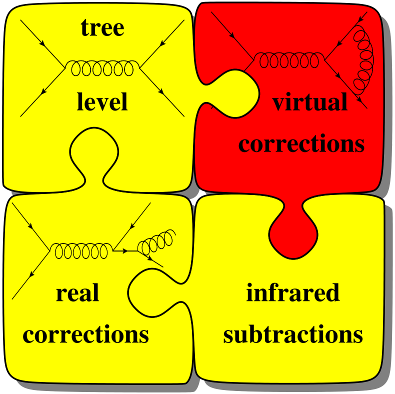


Complete NLO amplitude requires

- tree level amplitude,
- virtual corrections,
- real emission,
- subtraction terms
- Monte Carlo integrator, parton shower, PDFs, . . .
- . . . *but* the problem is nicely modular.

Golem and Samurai collaborations focus on virtual corrections

Anatomy of an NLO Calculation



Complete NLO amplitude requires

- tree level amplitude,
- virtual corrections,
- real emission,
- subtraction terms
- Monte Carlo integrator, parton shower, PDFs, ...
- ... *but* the problem is nicely modular.

Golem and Samurai collaborations focus on virtual corrections

The ‘Original’ Golem Method

Recipe

1. generate one-loop diagrams
2. carry out color algebra
3. close spinor strings by using suitable projectors (rather than squaring the amplitude)
4. carry out tensor reduction (into ‘Golem basis’)
5. express result in terms of Mandelstam variables
6. cancel as many Gram determinants as possible

- ✓ Method proved successful in calculations up to five legs.
- ✓ Results e.g. for $pp \to VVj$.
- ✓ Successful application leads to efficient code.

The 'Original' Golem Method

Recipe

1. generate one-loop diagrams
2. carry out color algebra
3. close spinor strings by using suitable projectors
   (rather than squaring the amplitude)
4. carry out tensor reduction (into 'Golem basis')
5. express result in terms of Mandelstam variables
6. cancel as many Gram determinants as possible

✓ Method proved successful in calculations up to five legs.

✓ Results e.g. for $pp \to VVj$.

✓ Successful application leads to efficient code.

The 'Original' Golem Method

Recipe

1. generate one-loop diagrams
2. carry out color algebra
3. close spinor strings by using suitable projectors (rather than squaring the amplitude)
4. carry out tensor reduction (into 'Golem basis')
5. express result in terms of Mandelstam variables
6. cancel as many Gram determinants as possible

- ✓ Method proved successful in calculations up to five legs.
- ✓ Results e.g. for $pp \to VVj$.
- ✓ Successful application leads to efficient code.

The 'Original' Golem Method

Recipe

1. generate one-loop diagrams
2. carry out color algebra
3. close spinor strings by using suitable projectors (rather than squaring the amplitude)
4. carry out tensor reduction (into 'Golem basis')
5. express result in terms of Mandelstam variables
6. cancel as many Gram determinants as possible

- ✗ Last step is ambiguous ('as many as possible').
- ✗ No guarantee to detect all cancellations.
- ✗ Difficult to automatize, computationally intense.

$\Rightarrow$ Purely algebraic approach is no option beyond $2 \to 3$.

The 'Original' Golem Method

Recipe

1. generate one-loop diagrams
2. carry out color algebra
3. close spinor strings by using suitable projectors (rather than squaring the amplitude)
4. carry out tensor reduction (into 'Golem basis')
5. express result in terms of Mandelstam variables
6. cancel as many Gram determinants as possible

- ✗ Last step is ambiguous ('as many as possible').
- ✗ No guarantee to detect all cancellations.
- ✗ Difficult to automatize, computationally intense.

$\Rightarrow$ Purely algebraic approach is no option beyond $2 \to 3$.

Problems in Diagrammatic One-Loop Calculations

- **Size:** #diagrams and #terms per diagram become large
  $\Rightarrow$ code size problematic in compiling and linking.
- **Speed:** #operations grows with code size.
- **Stability:**

  can Large cancellations between diagrams possible;
  lop loss of precision when #operations large;
  gra instabilities near Gram determinants.

Further design criteria for one-loop tools:

- **Flexibility:** Posibility to use different Models, Schemes etc.
- **Interoperability:** Integration with other programs/frameworks.
- **Usability:** Easy to learn, install and configure; documentation and support.

Problems in Diagrammatic One-Loop Calculations

- **Size:** #diagrams and #terms per diagram become large
  $\Rightarrow$ code size problematic in compiling and linking.
- **Speed:** #operations grows with code size.
- **Stability:**

  can Large cancellations between diagrams possible;
  lop loss of precision when #operations large;
  gra instabilities near Gram determinants.

Further design criteria for one-loop tools:

- **Flexibility:** Posibility to use different Models, Schemes etc.
- **Interoperability:** Integration with other programs/frameworks.
- **Usability:** Easy to learn, install and configure; documentation and support.

Problems in Diagrammatic One-Loop Calculations

- **Size:** #diagrams and #terms per diagram become large
  $\Rightarrow$ code size problematic in compiling and linking.
- **Speed:** #operations grows with code size.
- **Stability:**

  can Large cancellations between diagrams possible;
  lop loss of precision when #operations large;
  gra instabilities near Gram determinants.

Further design criteria for one-loop tools:

- **Flexibility:** Posibility to use different Models, Schemes etc.
- **Interoperability:** Integration with other programs/frameworks.
- **Usability:** Easy to learn, install and configure; documentation and support.

Problems in Diagrammatic One-Loop Calculations

- **Size:** #diagrams and #terms per diagram become large
  $\Rightarrow$ code size problematic in compiling and linking.
- **Speed:** #operations grows with code size.
- **Stability:**

  can Large cancellations between diagrams possible;
  lop loss of precision when #operations large;
  gra instabilities near Gram determinants.

Further design criteria for one-loop tools:

- **Flexibility:** Posibility to use different Models, Schemes etc.
- **Interoperability:** Integration with other programs/frameworks.
- **Usability:** Easy to learn, install and configure; documentation and support.

Improved Tensor Reduction (Golem95)

Numerical instabilities related to vanishing Gram determinants can hamper any muli-leg NLO calculation.

Gram determinants are introduced when reducing to scalar integrals. Thus, they are always present in on-shell methods.

Can we avoid introducing Gram determinants from the very beginning?

Passarino-Veltman Reduction & Gram Dets

$$\int \frac{\mathrm{d}^n k}{(2\pi)^n} \frac{p \cdot k}{(k+r_1)^2(k+r_2)^2(k+r_3)^2 k^2}$$

- If $\{r_1, r_2, r_3\}$ linearly independent:

$$p^\mu = \alpha_1 r_1^\mu + \alpha_2 r_2^\mu + \alpha_3 r_3^\mu + \alpha_\perp \epsilon^{\mu\nu\rho\sigma} r_{1\nu} r_{2\rho} r_{3\sigma}$$

- Since $2r_i \cdot k = (k+r_i)^2 - k^2 - r_i^2$
  $\Rightarrow$ decomposition into scalar integrals
- Need to solve: (which introduces Gram determinant)

$$\left(\begin{array}{ccc|c} r_1\cdot r_1 & r_1\cdot r_2 & r_1\cdot r_3 & 0 \\ r_2\cdot r_1 & r_2\cdot r_2 & r_2\cdot r_3 & 0 \\ r_3\cdot r_1 & r_3\cdot r_2 & r_3\cdot r_3 & 0 \\ \hline 0 & 0 & 0 & \det G \end{array}\right) \cdot \left(\begin{array}{c} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \hline \alpha_\perp \end{array}\right) = \left(\begin{array}{c} p\cdot r_1 \\ p\cdot r_2 \\ p\cdot r_3 \\ \hline \epsilon^{p r_1 r_2 r_3} \end{array}\right)$$

Passarino-Veltman Reduction & Gram Dets

$$\int \frac{\mathrm{d}^n k}{(2\pi)^n} \frac{p \cdot k}{(k+r_1)^2 (k+r_2)^2 (k+r_3)^2 k^2}$$

- If $\{r_1, r_2, r_3\}$ linearly independent:

$$p^\mu = \alpha_1 r_1^\mu + \alpha_2 r_2^\mu + \alpha_3 r_3^\mu + \alpha_\perp \epsilon^{\mu\nu\rho\sigma} r_{1\nu} r_{2\rho} r_{3\sigma}$$

- Since $2r_i \cdot k = (k+r_i)^2 - k^2 - r_i^2$
  $\Rightarrow$ decomposition into scalar integrals
- Need to solve: (which introduces Gram determinant)

$$\left(\begin{array}{ccc|c} r_1 \cdot r_1 & r_1 \cdot r_2 & r_1 \cdot r_3 & 0 \\ r_2 \cdot r_1 & r_2 \cdot r_2 & r_2 \cdot r_3 & 0 \\ r_3 \cdot r_1 & r_3 \cdot r_2 & r_3 \cdot r_3 & 0 \\ \hline 0 & 0 & 0 & \det G \end{array}\right) \cdot \left(\begin{array}{c} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \hline \alpha_\perp \end{array}\right) = \left(\begin{array}{c} p \cdot r_1 \\ p \cdot r_2 \\ p \cdot r_3 \\ \hline \epsilon^{p r_1 r_2 r_3} \end{array}\right)$$

Passarino-Veltman Reduction & Gram Dets

$$\int \frac{\mathrm{d}^n k}{(2\pi)^n} \frac{p \cdot k}{(k+r_1)^2(k+r_2)^2(k+r_3)^2 k^2}$$

- If $\{r_1, r_2, r_3\}$ linearly independent:

$$p^\mu = \alpha_1 r_1^\mu + \alpha_2 r_2^\mu + \alpha_3 r_3^\mu + \alpha_\perp \epsilon^{\mu\nu\rho\sigma} r_{1\nu} r_{2\rho} r_{3\sigma}$$

- Since $2r_i \cdot k = (k+r_i)^2 - k^2 - r_i^2$
  $\Rightarrow$ decomposition into scalar integrals
- Need to solve: (which introduces Gram determinant)

$$\left(\begin{array}{ccc|c} r_1\cdot r_1 & r_1\cdot r_2 & r_1\cdot r_3 & 0 \\ r_2\cdot r_1 & r_2\cdot r_2 & r_2\cdot r_3 & 0 \\ r_3\cdot r_1 & r_3\cdot r_2 & r_3\cdot r_3 & 0 \\ \hline 0 & 0 & 0 & \det G \end{array}\right) \cdot \left(\begin{array}{c} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \hline \alpha_\perp \end{array}\right) = \left(\begin{array}{c} p\cdot r_1 \\ p\cdot r_2 \\ p\cdot r_3 \\ \hline \epsilon^{p r_1 r_2 r_3} \end{array}\right)$$

Passarino-Veltman Reduction & Gram Dets

$$\int \frac{\mathrm{d}^n k}{(2\pi)^n} \frac{p \cdot k}{(k+r_1)^2(k+r_2)^2(k+r_3)^2 k^2}$$

- If $\{r_1, r_2, r_3\}$ linearly independent:

$$p^\mu = \alpha_1 r_1^\mu + \alpha_2 r_2^\mu + \alpha_3 r_3^\mu + \alpha_\perp \epsilon^{\mu\nu\rho\sigma} r_{1\nu} r_{2\rho} r_{3\sigma}$$

- Since $2r_i \cdot k = (k+r_i)^2 - k^2 - r_i^2$
  $\Rightarrow$ decomposition into scalar integrals
- Need to solve: (which introduces Gram determinant)

$$\left(\begin{array}{ccc|c} r_1\cdot r_1 & r_1\cdot r_2 & r_1\cdot r_3 & 0 \\ r_2\cdot r_1 & r_2\cdot r_2 & r_2\cdot r_3 & 0 \\ r_3\cdot r_1 & r_3\cdot r_2 & r_3\cdot r_3 & 0 \\ \hline 0 & 0 & 0 & \det G \end{array}\right) \cdot \left(\begin{array}{c} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \hline \alpha_\perp \end{array}\right) = \left(\begin{array}{c} p\cdot r_1 \\ p\cdot r_2 \\ p\cdot r_3 \\ \hline \epsilon^{p r_1 r_2 r_3} \end{array}\right)$$

Passarino-Veltman Reduction & Gram Dets

$$\int \frac{\mathrm{d}^n k}{(2\pi)^n} \frac{p \cdot k}{(k+r_1)^2 (k+r_2)^2 (k+r_3)^2 k^2}$$

- If $\{r_1, r_2, r_3\}$ linearly independent:

$$p^\mu = \alpha_1 r_1^\mu + \alpha_2 r_2^\mu + \alpha_3 r_3^\mu + \alpha_\perp \epsilon^{\mu\nu\rho\sigma} r_{1\nu} r_{2\rho} r_{3\sigma}$$

- Since $2r_i \cdot k = (k+r_i)^2 - k^2 - r_i^2$
  $\Rightarrow$ decomposition into scalar integrals
- Need to solve: (which introduces Gram determinant)

$$\left(\begin{array}{ccc|c} r_1 \cdot r_1 & r_1 \cdot r_2 & r_1 \cdot r_3 & 0 \\ r_2 \cdot r_1 & r_2 \cdot r_2 & r_2 \cdot r_3 & 0 \\ r_3 \cdot r_1 & r_3 \cdot r_2 & r_3 \cdot r_3 & 0 \\ \hline 0 & 0 & 0 & \det G \end{array}\right) \cdot \left(\begin{array}{c} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \hline \alpha_\perp \end{array}\right) = \left(\begin{array}{c} p \cdot r_1 \\ p \cdot r_2 \\ p \cdot r_3 \\ \hline \epsilon^{p r_1 r_2 r_3} \end{array}\right)$$

$$I_N^{d;\mu_1\ldots\mu_r}(S) = \int \frac{\mathrm{d}^d k}{i\pi^d/2} \frac{k^{\mu_1}k^{\mu_2}\cdots k^{\mu_r}}{[(k+r_1)^2-m_1^2]\cdots[(k+r_N)^2-m_N^2]}$$

$$S_{ij} = (r_i-r_j)^2-m_i^2-m_j^2$$

$$G_{ij} = 2r_i\cdot r_j$$

- Tensor Integrals from loop momentum

http://lappweb.in2p3.fr/lapth/Golem/golem95.html

Tensor Reduction: The Golem Method

$$I_N^{d;\mu_1\ldots\mu_r}(S) = \int \frac{\mathrm{d}^d k}{i\pi^d/2} \frac{k^{\mu_1} k^{\mu_2} \cdots k^{\mu_r}}{[(k+r_1)^2 - m_1^2] \cdots [(k+r_N)^2 - m_N^2]}$$

$$I_N^d(S) = (-1)^N \Gamma(N - d/2) \int_0 \mathrm{d}^N z \frac{\delta(1 - \sum z_i)}{\left(-\frac{1}{2} z^\intercal S z - i\delta\right)^{N-d/2}}$$

- Reduction to scalar basis $\Rightarrow (\det G)^{-1}$

http://lappweb.in2p3.fr/lapth/Golem/golem95.html

Tensor Reduction: The Golem Method

http://lappweb.in2p3.fr/lapth/Golem/golem95.html

$$I_N^{d;\mu_1\ldots\mu_r}(S) = \int\frac{\mathrm{d}^dk}{i\pi^d/2}\frac{k^{\mu_1}k^{\mu_2}\cdots k^{\mu_r}}{[(k+r_1)^2-m_1^2]\cdots[(k+r_N)^2-m_N^2]}$$

$$I_N^d(l_1,\ldots,l_p;S) = (-1)^N\Gamma(N-d/2)\int_0\mathrm{d}^Nz\frac{\delta(1-\sum z_i)z_{l_1}\cdots z_{l_p}}{\left(-\frac{1}{2}z^\intercal Sz-i\delta\right)^{N-d/2}}$$

$$I_N^d(S) = (-1)^N\Gamma(N-d/2)\int_0\mathrm{d}^Nz\frac{\delta(1-\sum z_i)}{\left(-\frac{1}{2}z^\intercal Sz-i\delta\right)^{N-d/2}}$$

- GOLEM basis: integrals with numerator

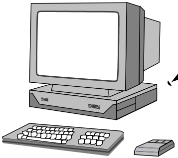
http://lappweb.in2p3.fr/lapth/Golem/golem95.html

Tensor Reduction: The Golem Method

$$I_N^{d;\mu_1\ldots\mu_r}(S) = \int\frac{\mathrm{d}^dk}{i\pi^d/2}\frac{k^{\mu_1}k^{\mu_2}\cdots k^{\mu_r}}{[(k+r_1)^2-m_1^2]\cdots[(k+r_N)^2-m_N^2]}$$

$$I_N^d(l_1,\ldots,l_p;S) = (-1)^N\Gamma(N-d/2)\int_0\mathrm{d}^Nz\frac{\delta(1-\sum z_i)z_{l_1}\cdots z_{l_p}}{\left(-\frac{1}{2}z^\mathsf{T}Sz-i\delta\right)^{N-d/2}}$$

$$I_N^d(S) = (-1)^N\Gamma(N-d/2)\int_0\mathrm{d}^Nz\frac{\delta(1-\sum z_i)}{\left(-\frac{1}{2}z^\mathsf{T}Sz-i\delta\right)^{N-d/2}}$$



- numerical evaluation or algebraic reduction

http://lappweb.in2p3.fr/lapth/Golem/golem95.html

golem95 One-Loop Integral Library

Current version of golem95

- http://lappweb.in2p3.fr/lapth/Golem/golem95.html
- algebraic separation of IR poles
- cache, avoiding multiple evaluation
- all required integrals for $N \leq 6$, massless & massive
- documentation, examples available

Under development:

- Numerical branch for massive integrals
- Complex propagator masses

Reduction at the Integrand Level (Samurai)

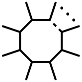
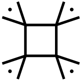
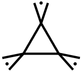
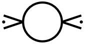
Most unitarity methods work in strictly four dimensions and have to distinguish between cut-constructible and rational terms.

Reduction at the integrand level can be combined with Feynman diagrammatic input giving access to many of the advantages of unitarity methods.

How can reduction at the integrand level be combined with integrands in n dimensions?

Reduction of the Integrand

- Analytic structure of the integrals implies structure of numerator. $(q^2 = \hat{q}^2 - \mu^2)$

 $= \sum d_{ijkl}$  $+ \sum c_{ijk}$  $+ \sum b_{ij}$  $+ \mathcal{R}$

$$= \int \mathrm{d}^n q \frac{N(\hat{q}, \mu^2)}{\prod_j [(q + r_j)^2 - m_j^2 + i\delta]}$$

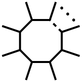
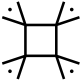
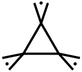
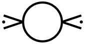
- Full reducibility to scalar integrals implies

$$N(\hat{q}, \mu^2) = \sum_{N,r} \sum_{j_1 \neq \ldots \neq j_N} \alpha^{(r)}_{j_1 \cdots j_N} \cdot (\mu^2)^r \prod_{j_i} [(\hat{q} + r_{j_i})^2 - m_{j_i}^2 + i\delta]$$

- Reconstruct $N(\hat{q}, \mu^2)$ by fitting $\alpha^{(r)}_{j_1 \cdots j_N}$ numerically at fixed values of $\hat{q}$ and μ^2

Reduction of the Integrand

- Analytic structure of the integrals implies structure of numerator. $(q^2 = \hat{q}^2 - \mu^2)$

 $= \sum d_{ijkl}$  $+ \sum c_{ijk}$  $+ \sum b_{ij}$  $+ \mathcal{R}$

$$= \int \mathrm{d}^n q \frac{N(\hat{q}, \mu^2)}{\prod_j [(q + r_j)^2 - m_j^2 + i\delta]}$$

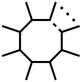
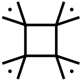
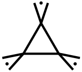
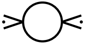
- Full reducibility to scalar integrals implies

$$N(\hat{q}, \mu^2) = \sum_{N,r} \sum_{j_1 \neq \ldots \neq j_N} \alpha^{(r)}_{j_1 \ldots j_N} \cdot (\mu^2)^r \prod_{j_i} [(\hat{q} + r_{j_i})^2 - m_{j_i}^2 + i\delta]$$

- Reconstruct $N(\hat{q}, \mu^2)$ by fitting $\alpha^{(r)}_{j_1 \ldots j_N}$ numerically at fixed values of $\hat{q}$ and μ^2

Reduction of the Integrand

- Analytic structure of the integrals implies structure of numerator. $(q^2 = \hat{q}^2 - \mu^2)$

 $= \sum d_{ijkl}$  $+ \sum c_{ijk}$  $+ \sum b_{ij}$  $+ \mathcal{R}$

$$= \int \mathrm{d}^n q \frac{N(\hat{q}, \mu^2)}{\prod_j [(q + r_j)^2 - m_j^2 + i\delta]}$$

- Full reducibility to scalar integrals implies

$$N(\hat{q}, \mu^2) = \sum_{N,r} \sum_{j_1 \neq \ldots \neq j_N} \alpha^{(r)}_{j_1 \ldots j_N} \cdot (\mu^2)^r \prod_{j_i} [(\hat{q} + r_{j_i})^2 - m_{j_i}^2 + i\delta]$$

- Reconstruct $N(\hat{q}, \mu^2)$ by fitting $\alpha^{(r)}_{j_1 \ldots j_N}$ numerically at fixed values of $\hat{q}$ and μ^2

Integrand Reduction with SAMURAI

- Four-dimensional integrand $N(\hat{q}, 0)$: only partial reconstruction of $\mathcal{R}$.
- SAMURAI fits $N(\hat{q}, \mu^2)$,
- works with loop Feynman-Diagrams or products of trees,
- can detect unstable points by fast reconstruction tests,
- runs in double and quadruple precision.

As part of `golem-2.0`

- reduction of code size (numerators vs. form factors),
- reliable numerical control through reconstruction tests (plus tests at the amplitude level),
- opens window to higher multiplicities ($N > 6$).

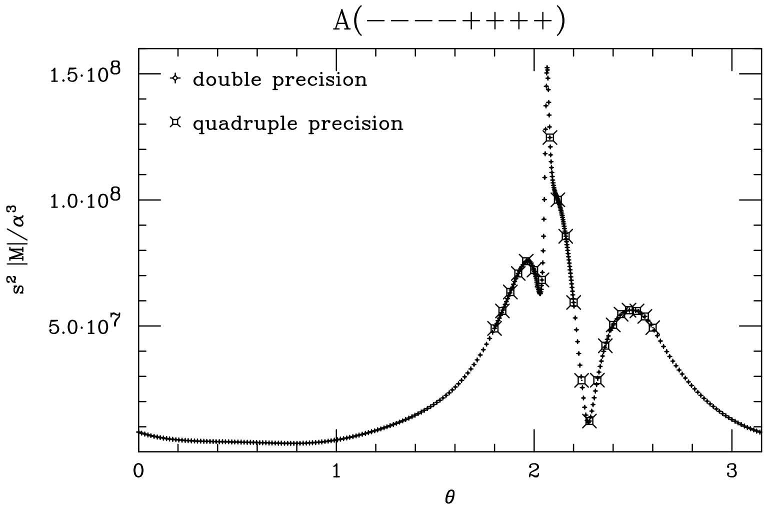
Integrand Reduction with SAMURAI

- Four-dimensional integrand $N(\hat{q}, 0)$: only partial reconstruction of $\mathcal{R}$.
- SAMURAI fits $N(\hat{q}, \mu^2)$,
- works with loop Feynman-Diagrams or products of trees,
- can detect unstable points by fast reconstruction tests,
- runs in double and quadruple precision.

As part of `golem-2.0`

- reduction of code size (numerators vs. form factors),
- reliable numerical control through reconstruction tests (plus tests at the amplitude level),
- opens window to higher multiplicities ($N > 6$).

Integrand Reduction with SAMURAI



Tensorial Reconstruction at the Integrand Level

The Golem reduction is robust and safe in all regions of phase space.

The Samurai reduction is very fast and numerator expression are very compact.

Can we take the best of both worlds and combine the methods?

Tensorial Reconstruction

$$\mathcal{N}(\hat{q},\mu^2) = \sum_{\alpha=0}^{\lfloor R/2 \rfloor} (\mu^2)^\alpha \mathcal{N}_\alpha(\hat{q}) = \sum_{\alpha=0}^{\lfloor R/2 \rfloor} (\mu^2)^\alpha \sum_{r=0}^{R-2\alpha} C^{(r,\alpha)}_{\mu_1\ldots\mu_r} \hat{q}^{\mu_1}\cdots\hat{q}^{\mu_r}$$

- *Idea*: determination of $C^{(r,\alpha)}_{\mu_1\ldots\mu_r}$ through sampling over $(\hat{q},\mu^2)$.
- result can be used in two ways:
  - multiplikation with tensor integrals yields full result,
  - take R.H.S. as input $N'(\hat{q},\mu^2)$ for integrand reduction.
- Advantages
  - $\hat{q}$ does not need to be on the cut, can be real;
  - to solve: system with fixed coefficients, always stable;
  - $N'(\hat{q},\mu^2)$ evaluates faster than original $N(\hat{q},\mu^2)$.
  - switch between reduction methods at runtime based on quality of reconstruction tests;
  - no need for quadruple precision, no recompilation.

Tensorial Reconstruction

$$\mathcal{N}(\hat{q},\mu^2) = \sum_{\alpha=0}^{\lfloor R/2 \rfloor} (\mu^2)^\alpha \mathcal{N}_\alpha(\hat{q}) = \sum_{\alpha=0}^{\lfloor R/2 \rfloor} (\mu^2)^\alpha \sum_{r=0}^{R-2\alpha} C^{(r,\alpha)}_{\mu_1\ldots\mu_r} \hat{q}^{\mu_1}\cdots\hat{q}^{\mu_r}$$

- *Idea*: determination of $C^{(r,\alpha)}_{\mu_1\ldots\mu_r}$ through sampling over $(\hat{q},\mu^2)$.
- result can be used in two ways:
  - multiplikation with tensor integrals yields full result,
  - take R.H.S. as input $N'(\hat{q},\mu^2)$ for integrand reduction.
- Advantages
  - $\hat{q}$ does not need to be on the cut, can be real;
  - to solve: system with fixed coefficients, always stable;
  - $N'(\hat{q},\mu^2)$ evaluates faster than original $N(\hat{q},\mu^2)$.
  - switch between reduction methods at runtime based on quality of reconstruction tests;
  - no need for quadruple precision, no recompilation.

Tensorial Reconstruction

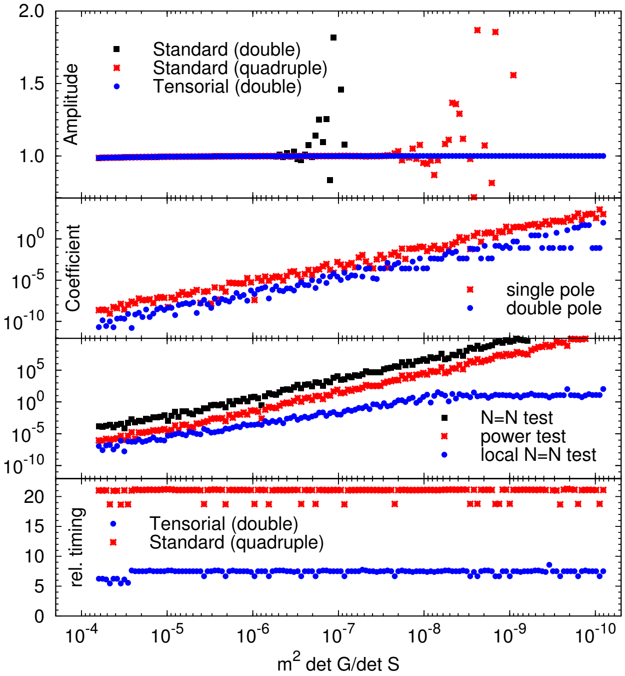
$$\mathcal{N}(\hat{q},\mu^2) = \sum_{\alpha=0}^{\lfloor R/2 \rfloor} (\mu^2)^\alpha \mathcal{N}_\alpha(\hat{q}) = \sum_{\alpha=0}^{\lfloor R/2 \rfloor} (\mu^2)^\alpha \sum_{r=0}^{R-2\alpha} C^{(r,\alpha)}_{\mu_1\ldots\mu_r} \hat{q}^{\mu_1}\cdots\hat{q}^{\mu_r}$$

- *Idea*: determination of $C^{(r,\alpha)}_{\mu_1\ldots\mu_r}$ through sampling over $(\hat{q},\mu^2)$.
- result can be used in two ways:
  - multiplikation with tensor integrals yields full result,
  - take R.H.S. as input $N'(\hat{q},\mu^2)$ for integrand reduction.
- Advantages
  - $\hat{q}$ does not need to be on the cut, can be real;
  - to solve: system with fixed coefficients, always stable;
  - $N'(\hat{q},\mu^2)$ evaluates faster than original $N(\hat{q},\mu^2)$.
  - switch between reduction methods at runtime based on quality of reconstruction tests;
  - no need for quadruple precision, no recompilation.

Tensorial Reconstruction

$$\mathcal{N}(\hat{q},\mu^2) = \sum_{\alpha=0}^{\lfloor R/2 \rfloor} (\mu^2)^\alpha \mathcal{N}_\alpha(\hat{q}) = \sum_{\alpha=0}^{\lfloor R/2 \rfloor} (\mu^2)^\alpha \sum_{r=0}^{R-2\alpha} C^{(r,\alpha)}_{\mu_1\ldots\mu_r} \hat{q}^{\mu_1} \cdots \hat{q}^{\mu_r}$$

- *Idea*: determination of $C^{(r,\alpha)}_{\mu_1\ldots\mu_r}$ through sampling over $(\hat{q},\mu^2)$.
- result can be used in two ways:
  - multiplikation with tensor integrals yields full result,
  - take R.H.S. as input $N'(\hat{q},\mu^2)$ for integrand reduction.
- Advantages
  - $\hat{q}$ does not need to be on the cut, can be real;
  - to solve: system with fixed coefficients, always stable;
  - $N'(\hat{q},\mu^2)$ evaluates faster than original $N(\hat{q},\mu^2)$.
  - switch between reduction methods at runtime based on quality of reconstruction tests;
  - no need for quadruple precision, no recompilation.



Amplitude
Standard (double)
Standard (quadruple)
Tensorial (double)
Coefficient
single pole
double pole
N=N test
power test
local N=N test
rel. timing
Tensorial (double)
Standard (quadruple)
m^2 det G/det S

Tensorial Reconstruction

Hybrid method for improved timing:

- Tensorial reconstruction on $N(\hat{q}, \mu^2)$ gives equivalent function $N'(\hat{q}, \mu^2)$.
- Samurai sampling of N' can be faster than that of N.

# Lines	Time ratio "hybrid"/standard	
N	Rank = 4	Rank = 6
1	1.3	1.6
10	1.1	1.4
100	0.51	0.85
1000	0.30	0.59
10000	0.27	0.55

Tensorial Reconstruction

Hybrid method for improved timing:

- Tensorial reconstruction on $N(\hat{q}, \mu^2)$ gives equivalent function $N'(\hat{q}, \mu^2)$.
- Samurai sampling of N' can be faster than that of N.

# Lines	Time ratio "hybrid"/standard	
N	Rank = 4	Rank = 6
1	1.3	1.6
10	1.1	1.4
100	0.51	0.85
1000	0.30	0.59
10000	0.27	0.55

Assembling the Golem (`golem-2.0`)

The virtual matrix element is just one piece in a complete NLO calculation.

Golem95 and SAMURAI *are components which need to be supplied with an amplitude representation in a suitable format.*

The generation of one-loop amplitude representations equipped with common interfaces to existing frameworks is the missing link for a general one-loop matrix element evaluation.

The ‘Original’ Golem Method

Recipe

1. generate one-loop diagrams
2. carry out color algebra
3. close spinor strings by using suitable projectors (rather than squaring the amplitude)
4. carry out tensor reduction (into ‘Golem basis’)
5. express result in terms of Mandelstam variables
6. cancel as many Gram determinants as possible

- ✗ Last step is ambiguous (‘as many as possible’).
- ✗ No guarantee to detect all cancellations.
- ✗ Difficult to automatize, computationally intense.

The Next-To-Original Golem Method

Recipe

1. generate one-loop diagrams
2. carry out color algebra
3. introduce form factor representation for tensor integrals
4. express Lorentz structure in terms of spinor products
5. tensor reduction is done numerically with Golem95

- ✓ No ambiguous or process dependent steps, easy to automatise
- ✓ It works: used in $q\bar{q} \to b\bar{b}b\bar{b}$
- ✗ Can lead to big expressions
  $\Rightarrow$ code generation slow, compilation difficult

The New SAMURAI-Golem Method

Recipe

1. generate numerators only for one-loop diagrams
2. carry out color algebra
3. express Lorentz structure in terms of spinor products
4. tensor reduction is done numerically with SAMURAI and/or Golem95

✓ Common input for different reduction methods, very flexible

✓ Compact expressions, code generation much faster

✓ Fast *and* stable reduction

✓ Works with any model and most schemes (currently only 't Hooft-Veltman with comm. γ_5 implemented)

Assembling the Golem

Assembling the Golem

spinney: Helicity-Spinors in Form

spinney: Helicity-Spinors in Form

- Problem: numerator algebra
  - Lorentz indices
  - Dirac algebra, Traces
- Good support in Form (Traces, Vectors, Tensors)
- GOLEM uses helicity projections
  - Need for helicity spinors
  - plus manipulations
- No direct support in Form

- spinney: Form library
- implementation of helicity spinors
- massive and massless
- includes rules for Majorana fermions
- implements 't Hooft-Veltman scheme

spinney: Helicity-Spinors in Form

- Problem: numerator algebra
  - Lorentz indices
  - Dirac algebra, Traces
- Good support in Form (Traces, Vectors, Tensors)
- GOLEM uses helicity projections
  - Need for helicity spinors
  - plus manipulations
- No direct support in Form

- spinney: Form library
- implementation of helicity spinors
- massive and massless
- includes rules for Majorana fermions
- implements 't Hooft-Veltman scheme

spinney: Helicity-Spinors in Form

```
vectors k1, k2, p3, p4, k3, k4;
indices mu, nu; symbol m;
#include- spinney.hh
local Amp = UbarSpb(k2) * Sm(mu) * USpa(k1) *
        d(mu, nu) *
        UbarSpb(p3, +1) * Sm(nu) * USpa(p4, -1);
#call LightConeDecomposition(p3, k3, k2, m)
#call LightConeDecomposition(p4, k4, k2, m)
#call tHooftAlgebra
#call SpCollect
#call SpContractMetrics
#call SpContract
#call SpOpen
id SpDenominator(m?) = 1/m;
print;
.end
```

spinney: Helicity-Spinors in Form

Output

```
Amp =
       - 2*Spa2(k1,k4)*Spb2(k2,k3);
```

$$\text{Amp} = -2\langle k_1 k_4\rangle[k_2 k_3]$$

- "in principle" suitable for numerical evaluation
- toy example $\to$ real world: expressions much larger
- expressions must be optimized
  - save function calls (most expensive)
  - save multiplications (expensive)
  - reduce complexity for compiler

spinney: Helicity-Spinors in Form

Output

```
Amp =
   - 2*Spa2(k1,k4)*Spb2(k2,k3);
```

$$\mathrm{Amp} = -2\langle k_1 k_4\rangle[k_2 k_3]$$

- "in principle" suitable for numerical evaluation
- toy example $\rightarrow$ real world: expressions much larger
- expressions must be optimized
  - save function calls (most expensive)
  - save multiplications (expensive)
  - reduce complexity for compiler

spinney: Helicity-Spinors in Form

Output

```
Amp =
      - 2*Spa2(k1,k4)*Spb2(k2,k3);
```

$$\mathrm{Amp} = -2\langle k_1 k_4\rangle[k_2 k_3]$$

- "in principle" suitable for numerical evaluation
- toy example → real world: expressions much larger
- expressions must be optimized
  - save function calls (most expensive)
  - save multiplications (expensive)
  - reduce complexity for compiler

spinney: Helicity-Spinors in Form

Output

```
Amp =
    - 2*Spa2(k1,k4)*Spb2(k2,k3);
```

$$\text{Amp} = -2\langle k_1 k_4\rangle[k_2 k_3]$$

- "in principle" suitable for numerical evaluation
- toy example $\rightarrow$ real world: expressions much larger
- expressions must be optimized
  - save function calls (most expensive)
  - save multiplications (expensive)
  - reduce complexity for compiler

haggies: An Optimizing Code Generator

haggies: An Optimizing Code Generator

GOALS

- on the *algebra* side:
  - save function calls (most expensive)
  - save multiplications (expensive)
  - reduce complexity for compiler
- on the *interface* side:
  - large class of input expressions
  - target independent (syntax, type system, ...)

HAGGIES:

- multivariate Horner scheme [Ceberio, Kreinovich 03]
- common subexpression elimination [Aho, Sethi, Ullman 86]
- common coefficient extraction [Gopalakrishnan, Kalla 09]
- economic variable allocation [Poletto et al. 97]
- built-in rule based type checker
- regex based syntax transformations

haggies: An Optimizing Code Generator

GOALS

- on the *algebra* side:
  - save function calls (most expensive)
  - save multiplications (expensive)
  - reduce complexity for compiler
- on the *interface* side:
  - large class of input expressions
  - target independent (syntax, type system, ...)

HAGGIES:

- multivariate Horner scheme [Ceberio, Kreinovich 03]
- common subexpression elimination [Aho, Sethi, Ullman 86]
- common coefficient extraction [Gopalakrishnan, Kalla 09]
- economic variable allocation [Poletto et al. 97]
- built-in rule based type checker
- regex based syntax transformations

haggies: An Optimizing Code Generator

GOALS

- on the *algebra* side:
  - save function calls (most expensive)
  - save multiplications (expensive)
  - reduce complexity for compiler
- on the *interface* side:
  - large class of input expressions
  - target independent (syntax, type system, ...)

HAGGIES:

- multivariate Horner scheme [Ceberio, Kreinovich 03]
- common subexpression elimination [Aho, Sethi, Ullman 86]
- common coefficient extraction [Gopalakrishnan, Kalla 09]
- economic variable allocation [Poletto et al. 97]
- built-in rule based type checker
- regex based syntax transformations

`haggies`: An Optimizing Code Generator

Examples: helicity amplitude: $gg \to gZZ$ $(+++++)$ [Binoth]

Box Coefficient $I_4^6(s_{12}, s_{14}, s_{35}, m_Z^2)$

	multiplications	additions
unoptimized	187,760	53,364
optimized	2,957	4,253
savings	98%	92%

Bubble Coefficient $I_2^n(s_{25})$

	multiplications	additions
unoptimized	4,256,184	2,026,768
optimized	65,832	125,001
savings	98%	94%

haggies: An Optimizing Code Generator

Examples: helicity amplitude: $gg \to gZZ$ $(+++++)$ [Binoth]

Box Coefficient $I_4^6(s_{12}, s_{14}, s_{35}, m_Z^2)$

	multiplications	additions
unoptimized	187,760	53,364
optimized	2,957	4,253
savings	98%	92%

Bubble Coefficient $I_2^n(s_{25})$

	multiplications	additions
unoptimized	4,256,184	2,026,768
optimized	65,832	125,001
savings	98%	94%

haggies: An Optimizing Code Generator

Examples: helicity amplitude: $gg \to gZZ$ $(+++++)$ [Binoth]

Box Coefficient $I_4^6(s_{12}, s_{14}, s_{35}, m_Z^2)$

	multiplications	additions
unoptimized	187,760	53,364
optimized	2,957	4,253
savings	98%	92%

Bubble Coefficient $I_2^n(s_{25})$

	multiplications	additions
unoptimized	4,256,184	2,026,768
optimized	65,832	125,001
savings	98%	94%

Problems Revisited

- **Size:** #diagrams and #terms per diagram become large
  $\Rightarrow$ code size problematic in compiling and linking.
- **Speed:** #operations grows with code size.
- **Stability:**

  can Large cancellations between diagrams possible;
  lop loss of precision when #operations large;
  gra instabilities near Gram determinants.

Problems Revisited

	Size	Speed	Stability can	Stability lop	Stability gra
Golem95	✗ (1)				✓
SAMURAI	✓	✓	✓		✗ (2)
Tensorial Reconstruction	✓ (1)	✓			✓ (2)
`haggies`	✓	✓		✓	
Improved Accumulation			✓		
$\sum$ `golem-2.0`	✓	✓	✓	✓	✓

Problems Revisited

	Size	Speed	Stability		
			can	lop	gra
Golem95	✗ (1)				✓
SAMURAI	✓	✓	✓		✗ (2)
Tensorial Reconstruction	✓ (1)	✓			✓ (2)
`haggies`	✓	✓		✓	
Improved Accumulation			✓		
$\sum$ `golem-2.0`	✓	✓	✓	✓	✓

Golem Status Report

- ✓ Support for Golem95 and Samurai as reduction 'engines'
- ✓ Algebraic manipulations with Form and `spinney`,
  code generation with `haggies`,
  all publicly available as open source!
- ✓ Import of model files from FeynRules and LanHep/CalcHep
- ✗ Not yet implemented: Binoth-Accord interface for communication with MC programs.
- ✓ *but* all ingredients are there $\Rightarrow$ should be ready soon.
- ✓ Program will be released after full $b\bar{b}b\bar{b}$ amplitude is validated.

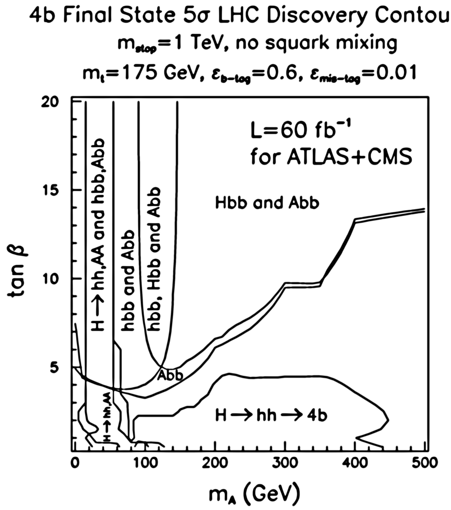
Results for $q\bar{q} \to b\bar{b}b\bar{b}$

The Golem framework combines algebraic methods for generating programs for the numerical evaluation of one-loop matrix elements.

The support of standardised interfaces facilitates working with user-defined models and using the generated code as plugin in existing Monte-Carlo programs.

Partial results have been obtained for the $pp \to b\bar{b}b\bar{b}$ amplitude. Recent improvements support the calculation of the full result.

$q\bar{q} \to b\bar{b}b\bar{b}$: An Important Background

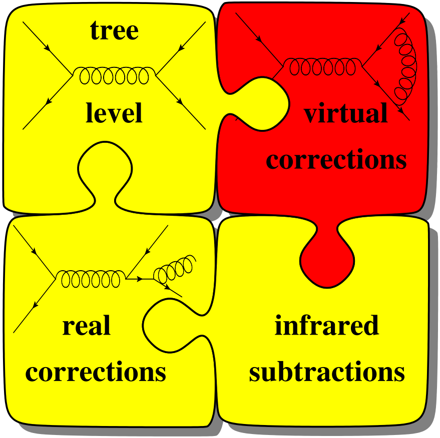


- Uncertainty on $b\bar{b}b\bar{b}$ crucial for BSM Higgs searches
- for certain MSSM scenarios: $H \to b\bar{b}b\bar{b}$ enhanced
- maybe only discovery channel
- also important for other BSM models

[Dai,Gunion,Vega]

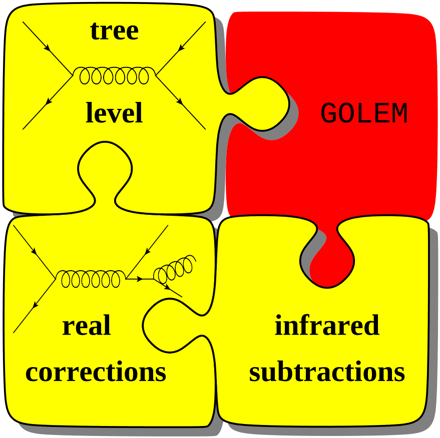
$q\bar{q} \to b\bar{b}b\bar{b}$: Setup

- virtual corrections: `GOLEM`
- Born part: `MadGraph`
  [F. Maltoni, T. Stelzer]
- real corrections: `MadGraph`
- subtractions: `MadDipole`
  [R. Frederix, T. Gehrmann, N. Greiner]
- integration/analysis (`MadEvent` [Maltoni, Stelzer])
- "plug and play": single subroutine call from `MadEvent` to `GOLEM`



$q\bar{q} \to b\bar{b}b\bar{b}$: Setup

- virtual corrections: GOLEM
- Born part: MadGraph
  [F. Maltoni, T. Stelzer]
- real corrections: MadGraph
- subtractions: MadDipole
  [R. Frederix, T. Gehrmann, N. Greiner]
- integration/analysis (MadEvent [Maltoni, Stelzer])
- "plug and play": single subroutine call from MadEvent to GOLEM

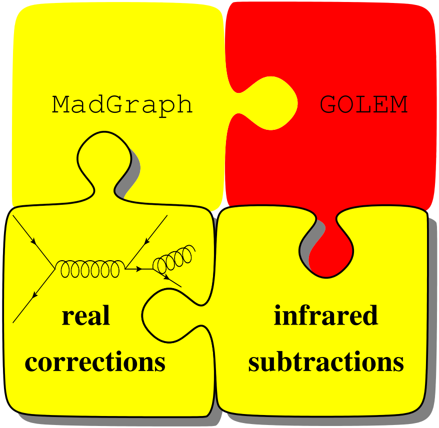


$q\bar{q} \to b\bar{b}b\bar{b}$: Setup

- virtual corrections: `GOLEM`
- Born part: `MadGraph`

  [F. Maltoni, T. Stelzer]
- real corrections: `MadGraph`
- subtractions: `MadDipole`

  [R. Frederix, T. Gehrmann, N. Greiner]
- integration/analysis (`MadEvent` [Maltoni, Stelzer])
- "plug and play": single subroutine call from `MadEvent` to `GOLEM`

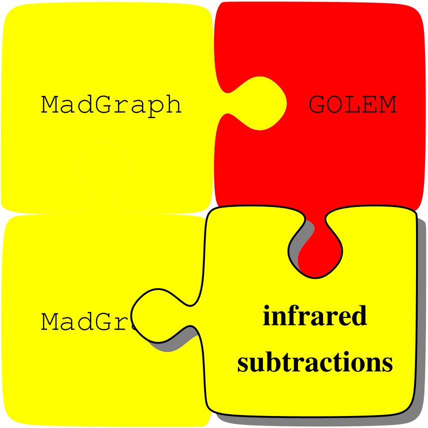


$q\bar{q} \to b\bar{b}b\bar{b}$: Setup

- virtual corrections: `GOLEM`
- Born part: `MadGraph`

  [F. Maltoni, T. Stelzer]
- real corrections: `MadGraph`
- subtractions: `MadDipole`

  [R. Frederix, T. Gehrmann, N. Greiner]
- integration/analysis (`MadEvent` [Maltoni, Stelzer])
- "plug and play": single subroutine call from `MadEvent` to `GOLEM`

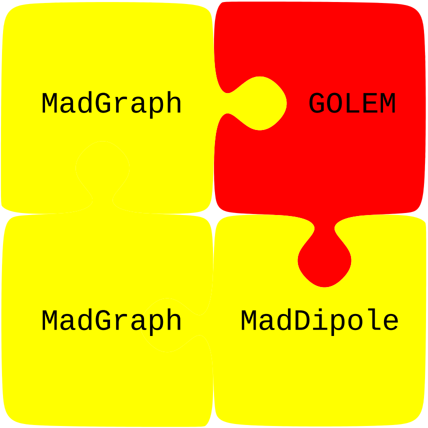


$q\bar{q} \to b\bar{b}b\bar{b}$: Setup

- virtual corrections: `GOLEM`
- Born part: `MadGraph`

  [F. Maltoni, T. Stelzer]
- real corrections: `MadGraph`
- subtractions: `MadDipole`

  [R. Frederix, T. Gehrmann, N. Greiner]
- integration/analysis (`MadEvent` [Maltoni, Stelzer])
- "plug and play": single subroutine call from `MadEvent` to `GOLEM`

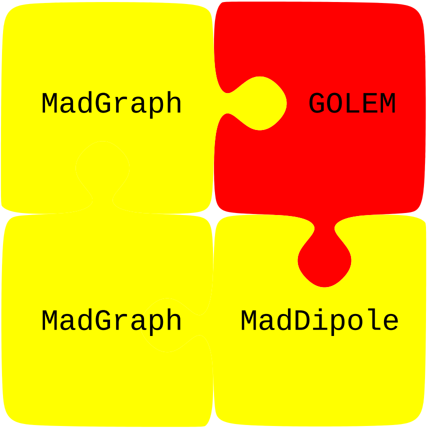


$q\bar{q} \to b\bar{b}b\bar{b}$: Setup

- virtual corrections: `GOLEM`
- Born part: `MadGraph`

  [F. Maltoni, T. Stelzer]
- real corrections: `MadGraph`
- subtractions: `MadDipole`

  [R. Frederix, T. Gehrmann, N. Greiner]
- integration/analysis (`MadEvent` [Maltoni, Stelzer])
- "plug and play": single subroutine call from `MadEvent` to `GOLEM`

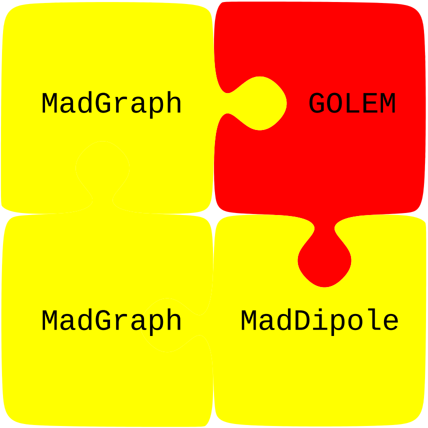


$q\bar{q} \to b\bar{b}b\bar{b}$: Setup

- virtual corrections: `GOLEM`
- Born part: `MadGraph`

  [F. Maltoni, T. Stelzer]
- real corrections: `MadGraph`
- subtractions: `MadDipole`

  [R. Frederix, T. Gehrmann, N. Greiner]
- integration/analysis (`MadEvent` [Maltoni, Stelzer])
- "plug and play": single subroutine call from `MadEvent` to `GOLEM`



$q\bar{q} \to b\bar{b}b\bar{b}$: Alternative Setup

- Born part, real emission and IR-subtractions: Whizard [Kilian,Moretti,Ohl,Reuter]
- virtual part: stand-alone `GOLEM`
- reweighting of unweighted LO events

$$\sigma = \frac{\sigma_{LO}}{|U|} \sum_{u \in U} \left(\frac{\mathrm{d}\sigma_{virt}(u)}{\mathrm{d}\sigma_{LO}(u)} + 1 \right)$$

- very efficient integration method

$q\bar{q} \to b\bar{b}b\bar{b}$: Checks

- LO contribution and real emission
  - two independent calculations:
    - MadEvent/MadGraph
    - Whizard
- IR-subtractions
  - two independent calculations:
    - MadDipole
    - implementation in Whizard
  - cut-off independence (α_{Nagy})
- virtual corrections
  - two independent implementations:
    - `GOLEM-2.0` (QGraf, Form, Fortran/`golem95`)
    - FeynArts/FeynCalc, Form, algebraic reduction
  - cancellation of poles in $1/(d-4)$
  - symmetry properties of the amplitude

$q\bar{q} \to b\bar{b}b\bar{b}$: Checks

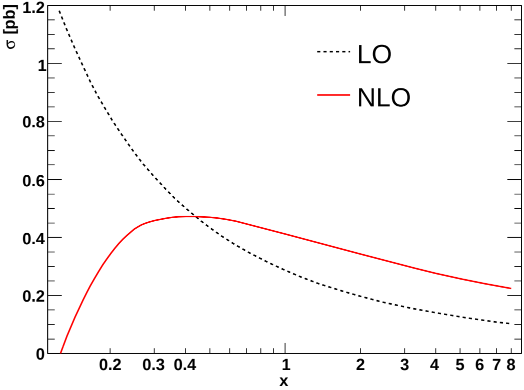
- LO contribution and real emission
  - two independent calculations:
    - MadEvent/MadGraph
    - Whizard
- IR-subtractions
  - two independent calculations:
    - MadDipole
    - implementation in Whizard
  - cut-off independence (α_{Nagy})
- virtual corrections
  - two independent implementations:
    - `GOLEM-2.0` (QGraf, Form, Fortran/`golem95`)
    - FeynArts/FeynCalc, Form, algebraic reduction
  - cancellation of poles in $1/(d-4)$
  - symmetry properties of the amplitude

$q\bar{q} \to b\bar{b}b\bar{b}$: Checks

- LO contribution and real emission
  - two independent calculations:
    - MadEvent/MadGraph
    - Whizard
- IR-subtractions
  - two independent calculations:
    - MadDipole
    - implementation in Whizard
  - cut-off independence (α_{Nagy})
- virtual corrections
  - two independent implementations:
    - `GOLEM-2.0` (QGraf, Form, Fortran/`golem95`)
    - FeynArts/FeynCalc, Form, algebraic reduction
  - cancellation of poles in $1/(d-4)$
  - symmetry properties of the amplitude

$q\bar{q} \to b\bar{b}b\bar{b}$: Results

$$\mu_R = x\mu_0$$

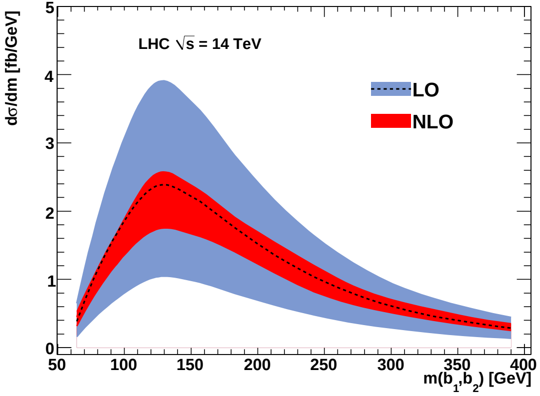


- significant reduction of scale dependence
- plateau region around $x = 0.5$
- stabilization of result
- error bands: $\mu_0/4 < \mu_R < 2\mu_0$
- complete analysis after inclusion of all channels

$\sqrt{s} = 14\,\text{TeV}$, $\mu_0 = \sqrt{\sum_j p_T^2(b_j)}$, $\mu_F = 100\,\text{GeV}$, $m_b = 0$

$q\bar{q} \to b\bar{b}b\bar{b}$: Results

m_{bb} of leading b-jets



$\sqrt{s} = 14$ TeV, $\mu_R = \frac{1}{2}\sqrt{\sum_j p_T^2(b_j)}$, $\mu_F = 100$ GeV, $m_b = 0$

- significant reduction of scale dependence
- plateau region around $x = 0.5$
- stabilization of result
- error bands: $\mu_0/4 < \mu_R < 2\mu_0$
- complete analysis after inclusion of all channels

Outlook and Summary

Even a good program can be improved.

The bottleneck in `golem-2.0` is the code generation phase, concerning both time and size.

The target is to skip code generation without compromising other achievements.

Can we find recursion relations for generating numerically the integrands of one-loop Feynman diagrams in n dimensions?

Outlook

At tree-level:

- Scalarization of Feynman rules

$$\Delta_{AB}^{(2s+1)}(p) = \sum_{\lambda}^{2s+1} \frac{\epsilon_{\lambda}^{A}(p)[\epsilon_{\lambda}^{B}(p)]^{*}}{p^2 - m^2}$$

- Implementations exist, e.g. helac-library
- Together with recursion relations
  $\Rightarrow$ efficient fully numerical implementation of amplitudes

At One-Loop level

✓ Recursion relations known, e.g. [v. Hameren]

✓ can be used for construction of $N(q)$

✗ Extension of scalar Feynman rules to $n \neq 4$ dimensions
$\rightarrow$ open research question for non-integer n...

Outlook

At tree-level:

- Scalarization of Feynman rules

$$\Delta_{AB}^{(2s+1)}(p) = \sum_{\lambda}^{2s+1} \frac{\epsilon_\lambda^A(p)[\epsilon_\lambda^B(p)]^*}{p^2 - m^2}$$

- Implementations exist, e.g. helac-library
- Together with recursion relations
  $\Rightarrow$ efficient fully numerical implementation of amplitudes

At One-Loop level

✓ Recursion relations known, e.g. [v. Hameren]

✓ can be used for construction of $N(q)$

✗ Extension of scalar Feynman rules to $n \neq 4$ dimensions
$\rightarrow$ open research question for non-integer n...

Summary

- Golem reduction method solves det G problem
- Golem95 for massive and massless integrals
- SAMURAI reduction at the integrand level
  - n-dim. numerators for full reconstruction of rat. terms
  - reliable reconstruction tests
- Improvements and combination of Golem95 and SAMURAI with tensorial reconstruction
- `spinney` for helicity spinors in Form
- efficient optimising code generation with `haggies`
- Golem95, `Samurai`, `spinney`, `haggies` and Form publicly available and open source
- `golem-2.0` new interface for FeynRules
- Binoth accord interface soon
- $gg \to b\bar{b}b\bar{b}$ to be validated
- `golem-2.0` release afterwards